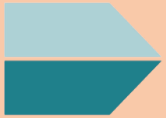
Cloud System Management

Cybersecurity

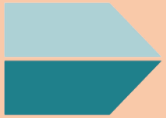
Cloud Security Day 2

Class Objectives

By the end of today's class, you will be able to:



Access your entire VNet from your jump box.



Install and run containers using Docker.

Set up Ansible connections to VMs inside your VNet.


Make sure you’re using your personal Azure accounts today — not the cyberxsecurity accounts.

Cloud Computing Recap

In the previous class, we covered:

01

The different cloud services and the *aaS acronyms.

02

The unique challenges of securing cloud deployments and building security from the ground up.

03

How to set up a virtual network protected by a firewall with a virtual machine on that network.

At this time, the VM is still inaccessible because our firewall is blocking traffic to it.

Today's Class

Today, we'll cover:

01 SSH connections and security group rules.

02 Containers, what they are, and their role in IT infrastructure.

03 Provisioners and their role in the larger concept of infrastructure as code.

04 Using Ansible to create infrastructure.

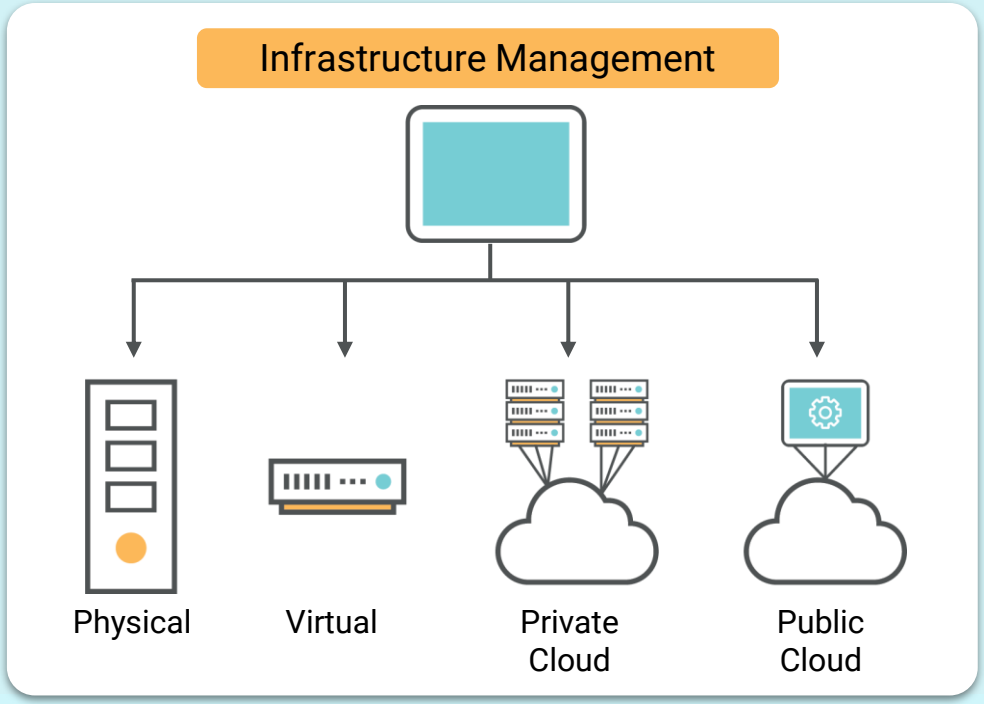
05 An introduction to network architecture and secure network design.

Containers, IaC, and Provisioners

Containers, IaC, and Provisioners

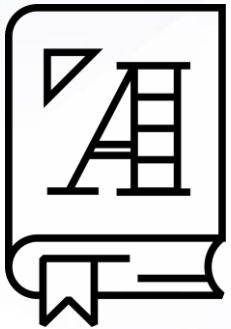
Containers, infrastructure as code, and provisioners are new technologies that provide powerful solutions to some of the most difficult problems of infrastructure management.

- Not all organizations leverage these tools fully.
- But, nearly all organizations are aware of their value, and many are updating their processes and toolkits to make better use of them.



Containers, IaC, and Provisioners

Potential employers will consider knowledge of containers, infrastructure as code, and automated provisioning valuable for both cloud-specific and "standard" systems/network administration roles.



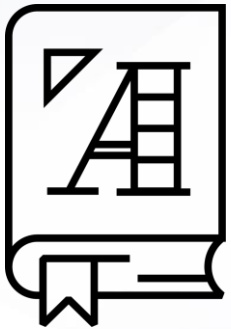
Containers can be thought of as “lightweight VMs.”

High-Level Overview: Containers

Containers are smaller than VMs (megabytes rather than gigabytes) and require fewer CPU resources.

- Because they are smaller, they can be downloaded and distributed more easily.
- Since they’re cheap, more of them can be run.
- They're also easier and faster to destroy and redeploy as needed.

(Google Containers)



Provisioners are tools that automatically configure VMs or containers for you.

High-Level Overview: Provisioners

Instead of manually logging into a machine and issuing commands like `apt get`, or editing configuration files yourself, you can use a provisioner to do this automatically.

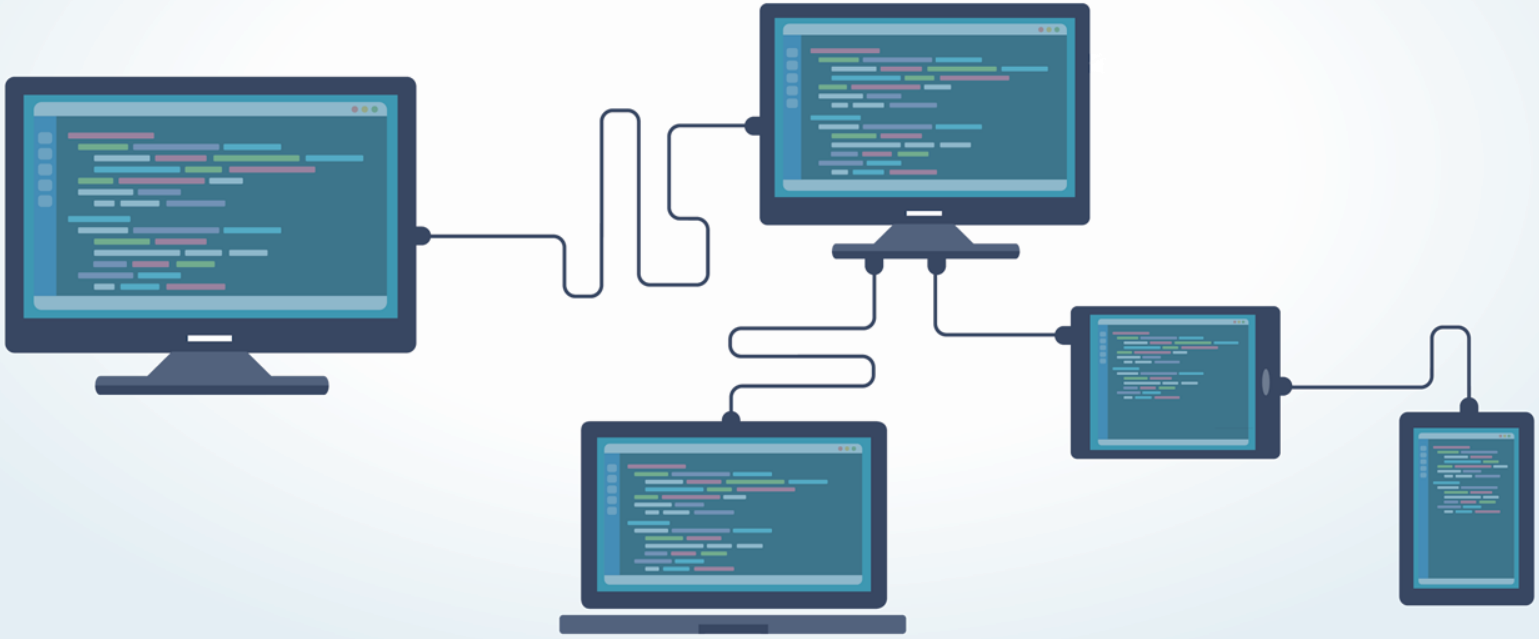
- This drastically reduces the potential for human error and simplifies the process of configuring potentially thousands of machines identically all at once.

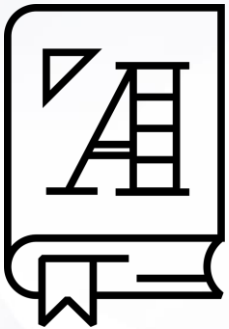
(Source)

Infrastructure as code (IaC) is the idea that the configurations for all of the VMs, containers, and networks in your deployment should be defined in text files, which you can use with provisioners to automatically recreate machines and networks whenever necessary.

High-Level Overview: Infrastructure as Code

The primary benefit to IaC is that everyone can see exactly how the network is configured by reading text files. These can be easily version controlled with tools like Git, Apple Time Machine, or Microsoft OneDrive.





Continuous Integration/ Continuous Deployment (CI/CD) is the concept of automatically updating machines on your network whenever your IaC files change.

High-Level Overview: Infrastructure as Code

Whenever you change a machine's configuration file:

Continuous Integration (CI) ensures that a new version of that machine is built immediately.

Continuous Deployment (CD) ensures that this new version is automatically deployed to your live environment.

The primary advantage to CI/CD is that it allows you to manage your entire network simply by updating IaC text files.

Foundations of Network Architecture

Remember: Our virtual machine is not accessible because our security group firewall is blocking all traffic to it.

This is to ensure that no one can possibly gain access to the VM.

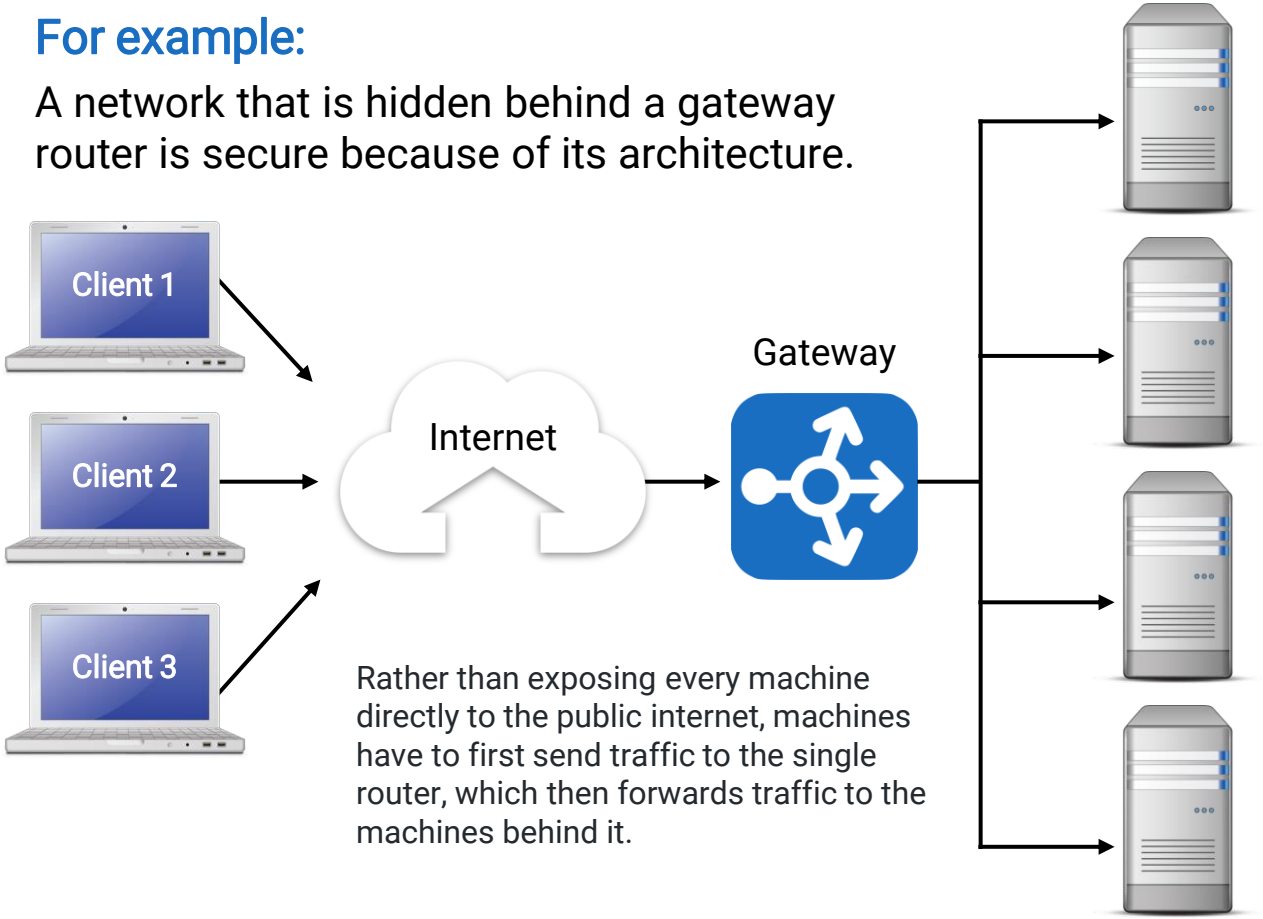
Secure Configuration and Secure Architecture

This is an example of “secure configuration,” as opposed to “secure architecture.”

	Secure Configuration	Secure Architecture
Ensures that….	An individual VM or network is protected from intrusion using well-considered rules, such as access control policies and firewall rules.	A poorly configured or malfunctioning individual machine can only cause a limited amount of damage.
Is secure because...	It follows the right rules.	It is “structurally sound.” In other words: Secure architecture deters and contains the effects of a breach, ensuring that even insecure machines are hard to compromise, and that damage to a single machine doesn't take down the entire network.

For example:

A network that is hidden behind a gateway router is secure because of its architecture.

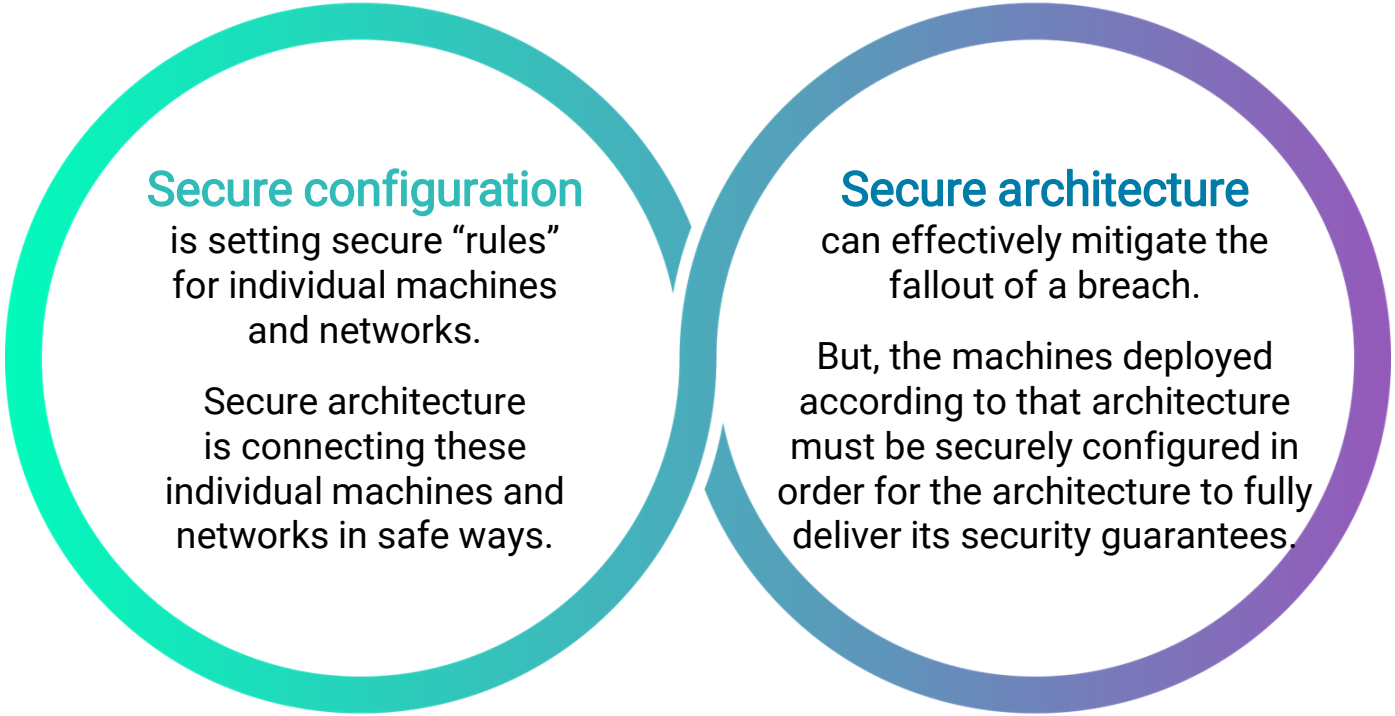


Rather than exposing every machine directly to the public internet, machines have to first send traffic to the single router, which then forwards traffic to the machines behind it.

This allows security personnel to implement strong access controls on a single machine (the router) rather than on every individual VM, drastically reducing the attack surface.

Secure Configuration and Secure Architecture

While secure configuration and secure architecture promote security in different ways, they must work together.



Network Redundancy

Network Redundancy

Important cloud security concepts include **fault tolerance** and **redundancy.**

- A **fault tolerant system** can keep running even if one or more components within the system fail.
- These systems continue to run because of **redundancy**: creating duplications of critical components or systems.
- If one system or component is lost or compromised, a redundant system or component can step in and keep the system going.
- From a security standpoint, this directly affects the reliability and availability of a system.

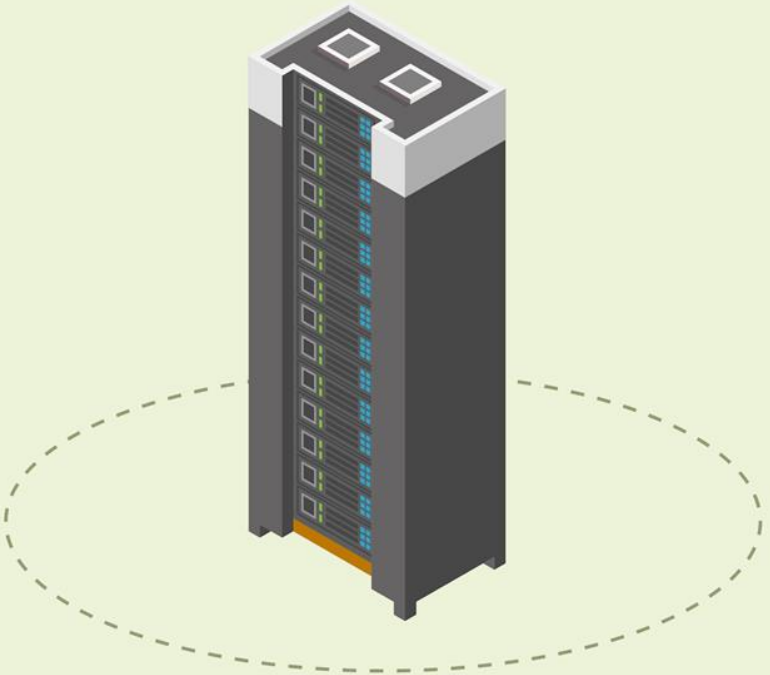
When making decisions about network architecture, we should always consider the amount of redundancy needed versus the amount of budget available.

Redundancy vs. Budget

Creating redundant systems for everything may be advisable from an engineering standpoint, but not every administrator will have the budget to do so.

- If Amazon was only running on one server, and the server went down, it would likely lose millions of dollars during downtime.
- It wouldn't make sense for a smaller web company to invest in the infrastructure needed to support a site like Amazon.

  The company would likely go out of business due to unsustainable operational costs.



So where does the cloud come in?

Cloud services allow a company to add resources as needed, scaling infrastructure as the business grows and only paying for what they need.

Activity: Cloud Architecture

In this activity, you will decide which network architectures work best for each given scenario.

Suggested Time:
15 Minutes

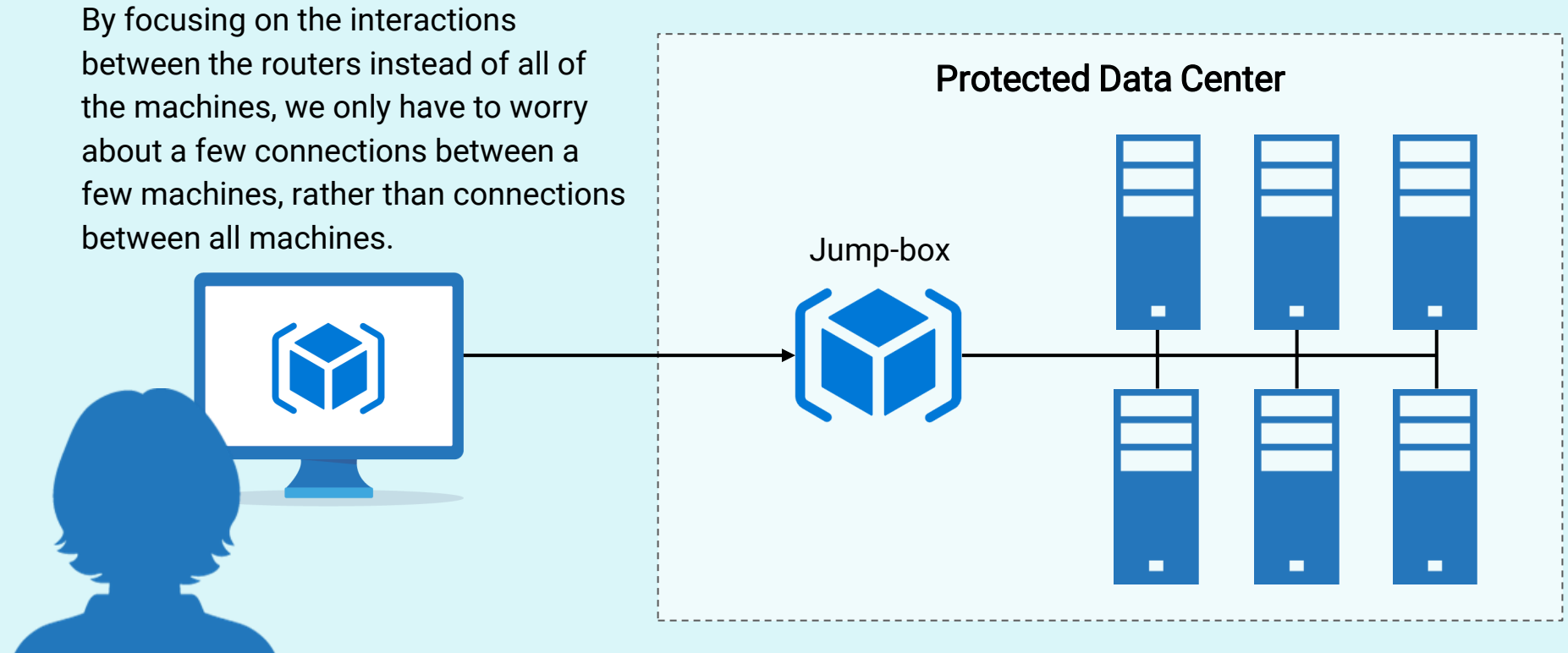
Jump Box Administration

Now, we’ll apply secure architecture methods to our jump box configurations.

Fanning In

Placing a gateway router between VMs on a network forces all traffic through a single node. Securing and monitoring this single node is called **fanning in**.

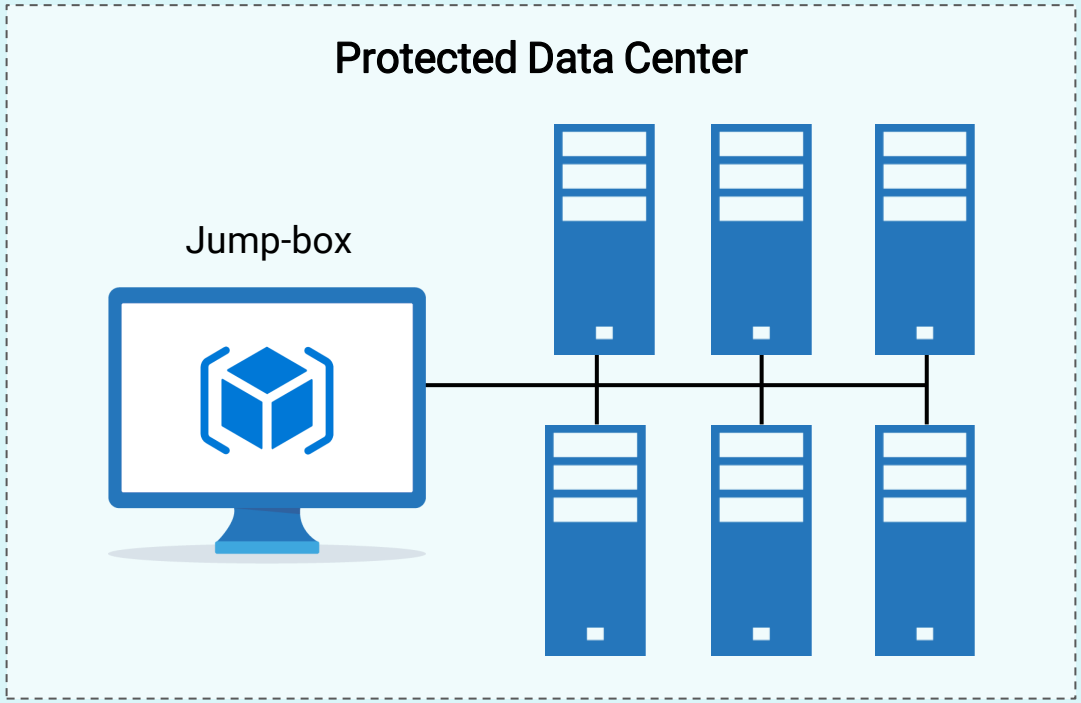
By focusing on the interactions between the routers instead of all of the machines, we only have to worry about a few connections between a few machines, rather than connections between all machines.



Fanning In

In this diagram, a jump box is essentially identical to a gateway router.

- The jump box is exposed to the public internet. In the classroom, we will be able to connect to the jump box's SSH port (22).
- It sits in front of other machines that are not exposed to the public internet.
- It controls access to the other machines by allowing connections from specific IP addresses and forwarding to those machines.



Further Steps

While this architecture is secure enough, we can and should further harden setups by:

01 Limiting the number of machines that our jump box can access.

02 Locking the root account and limiting **sudo** access of the admin account on the jump box.

03 Implementing log monitoring on the jump box.

04 Implementing two-factor authentication for SSH login to the jump box.

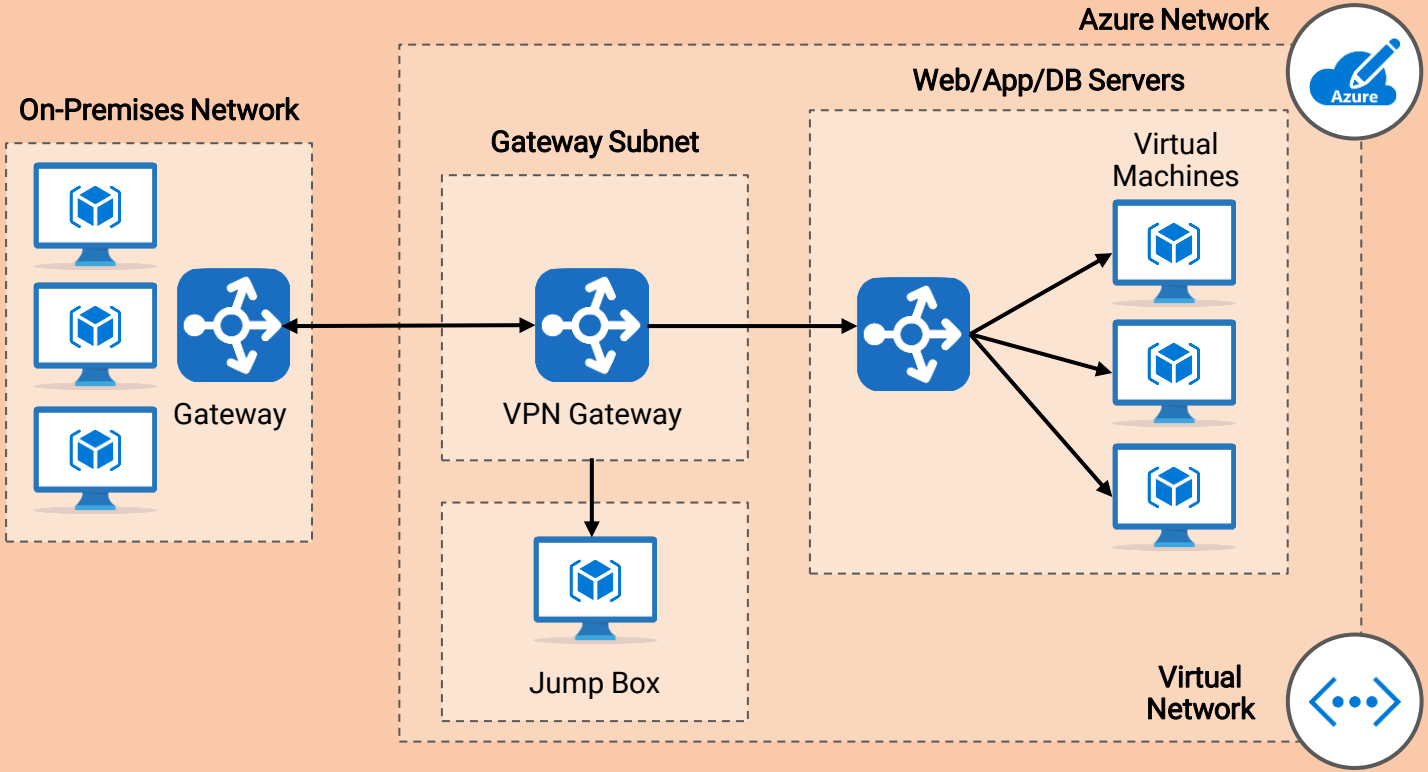
05 Implementing a host firewall (UFW or IPtables) on the jump box.

06 Limiting jump box network access with a virtual private network (VPN).

We can configure a virtual private network (VPN) to securely connect to a cloud network.

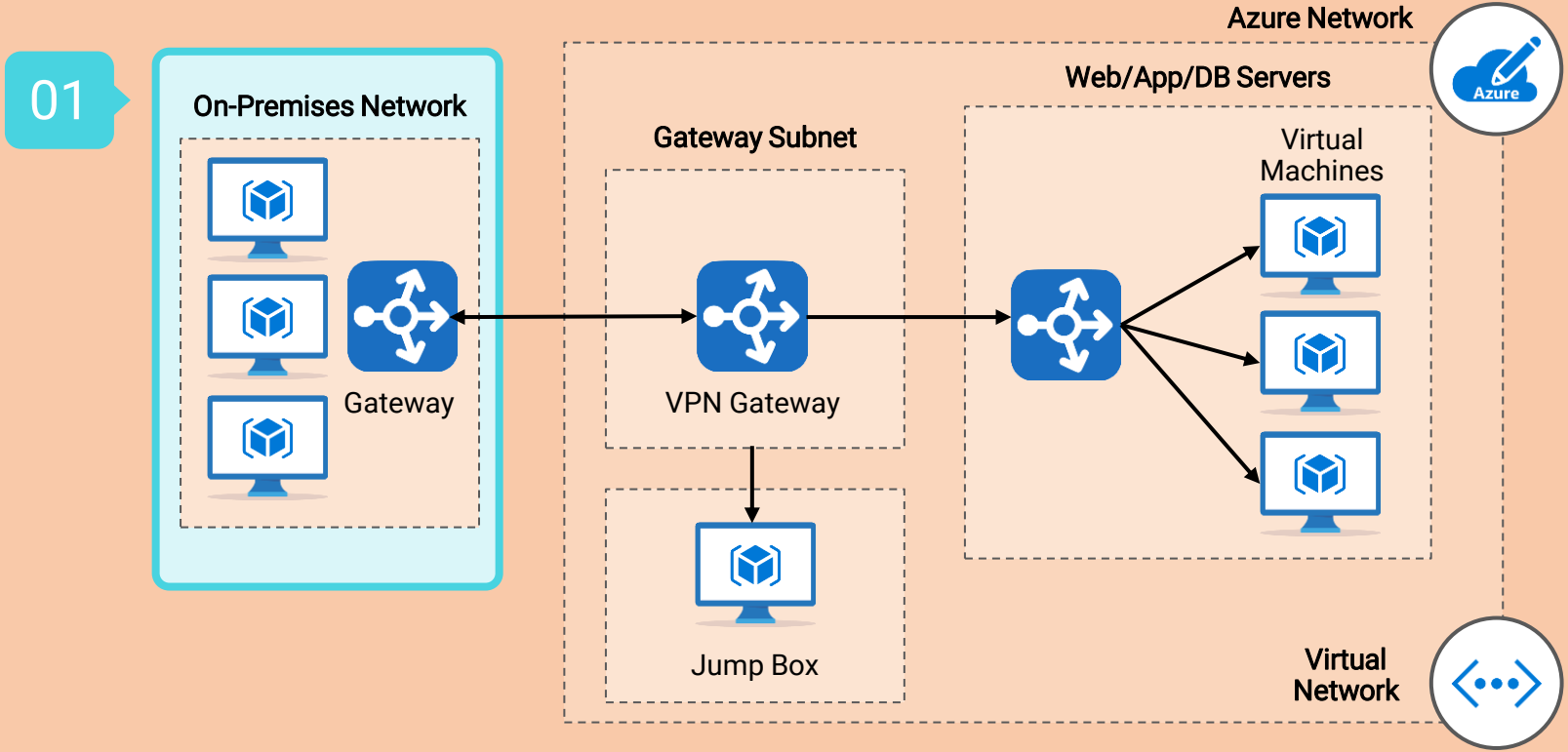
Virtual Private Networks

A virtual private network (VPN) creates a direct connection between your local network and a remote network.



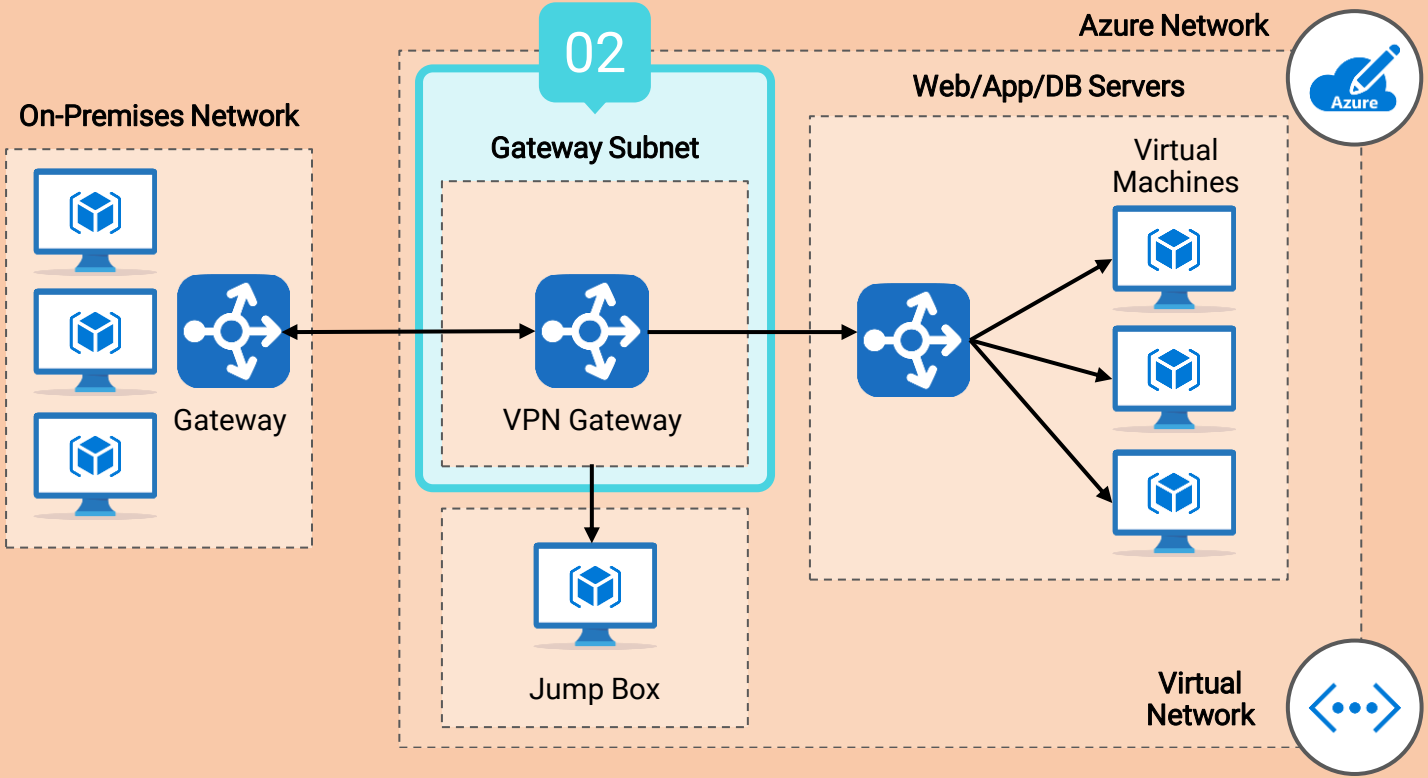
Virtual Private Networks

A VPN can encrypt all network traffic between your current network and your remote network.



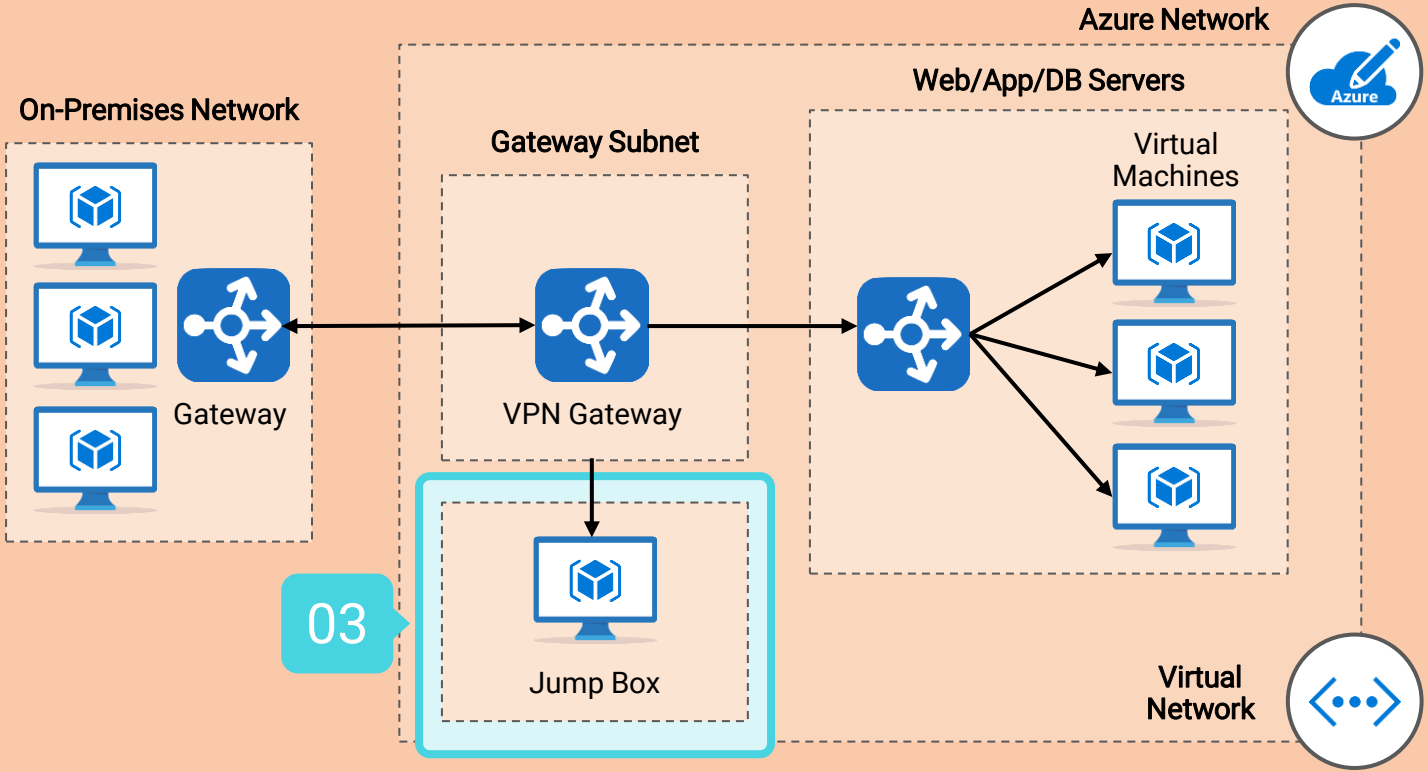
Virtual Private Networks

Once you are connected to the VPN, you have full access to all resources on the remote network, as if you were locally connected.



Virtual Private Networks

Many companies use VPNs to allow remote workers access to computers and servers that would otherwise only be accessible from the local network.



Activity: Jump Box Administration

In this activity, you will create a security group rule to allow SSH connections *only* from your current IP address and connect to your new virtual machine for management.

Suggested Time:
15 Minutes

Containers

Containers

Configuring machines may require downloading and installing different applications to work together. For example, when configuring a LAMP web server, you will need to install:

Linux

A web server like Apache

A database like MYSQL

A back-end codebase like PHP

For each of these items to work together, they need to be properly configured, which can be a time consuming process.

Containers

We can use VMs to configure a single LAMP server by:

- Setting up a VM as a LAMP server.
- Capturing an image of this VM.
- Duplicating this image whenever a new LAMP server is needed.

Containers

However, this method is “heavy.” VMs are large and take a long time to download and deploy.

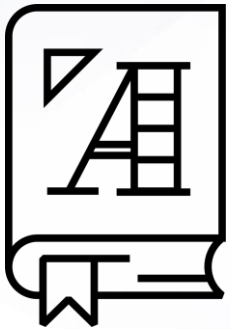
Additionally, if you clone an entire VM, most of the VM is “wasted space.”

Only a few files on the entire disk are actually relevant to running the LAMP server.

The rest are just operating system files.

Containers

It is more efficient if web servers share common resources like the OS kernel and core libraries, and are only individually responsible for maintaining what is different between them, such as the data in their databases.

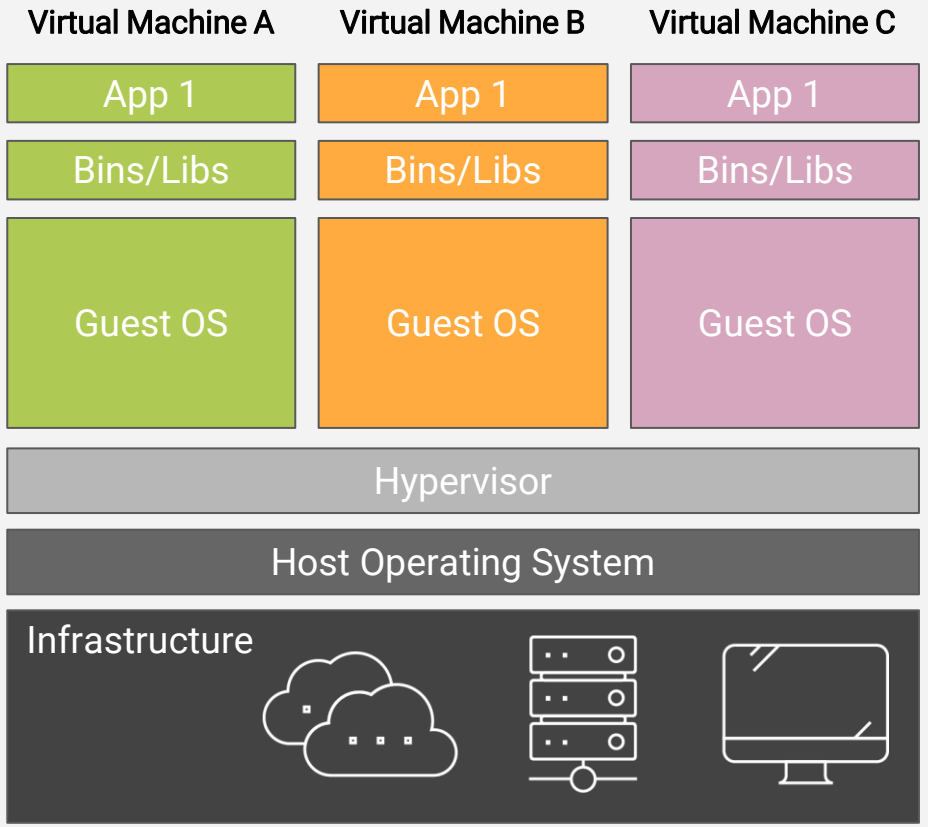


Containers are essentially lightweight VMs. These act as VMs but are smaller and use fewer resources by sharing the resources they have in common with other containers.

Containers vs. VMs

Like VMs, containers are simulated machines that run on a single host. However, two separate VMs running on the same host are completely independent of each other.

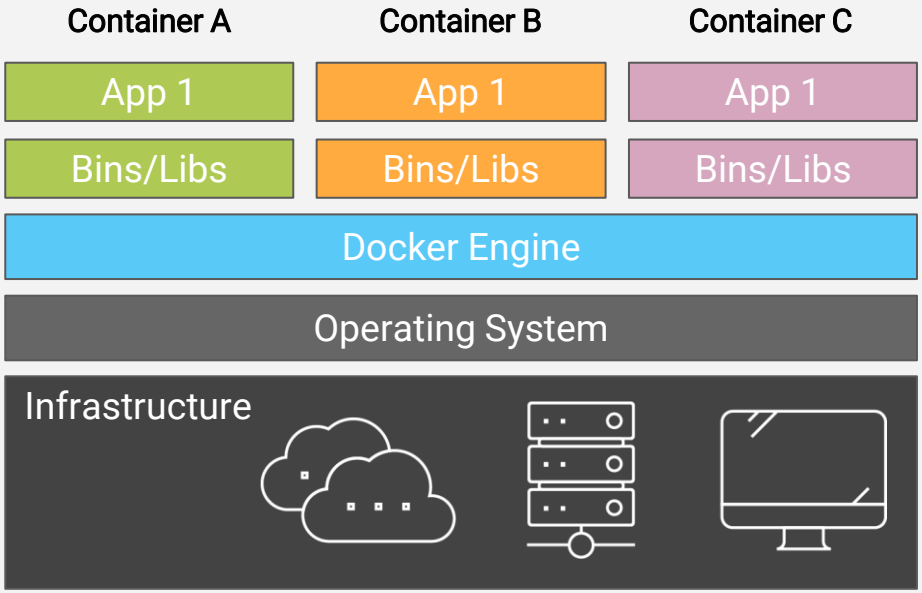
Virtual Machine A has no knowledge of Virtual Machine B, even if Virtual Machine A and B were created from identical images.



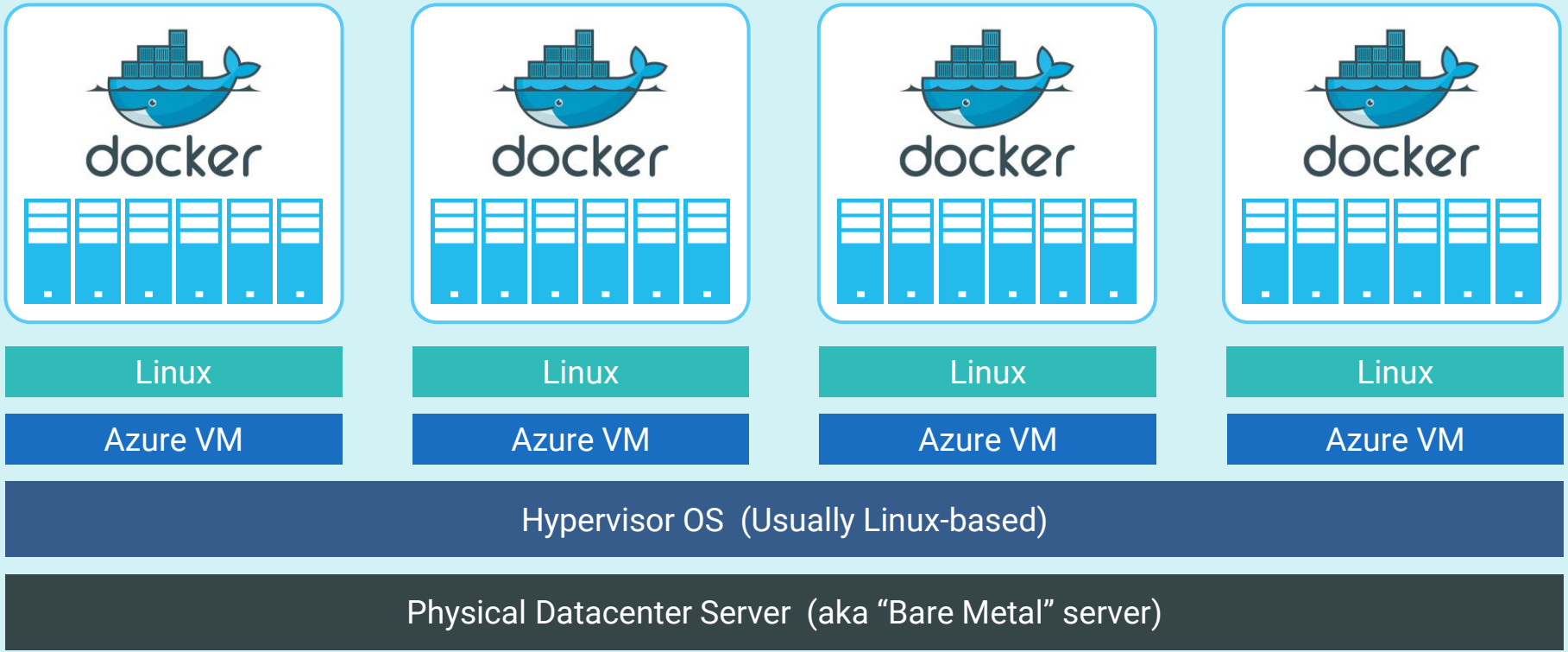
Containers vs. VMs

But, containers can share certain files. Container A, Container B, and Container C can run on a single host and use the same kernel.

They share these files and therefore only have to individually contain the files relevant to their applications, such as the LAMP server.



Containers

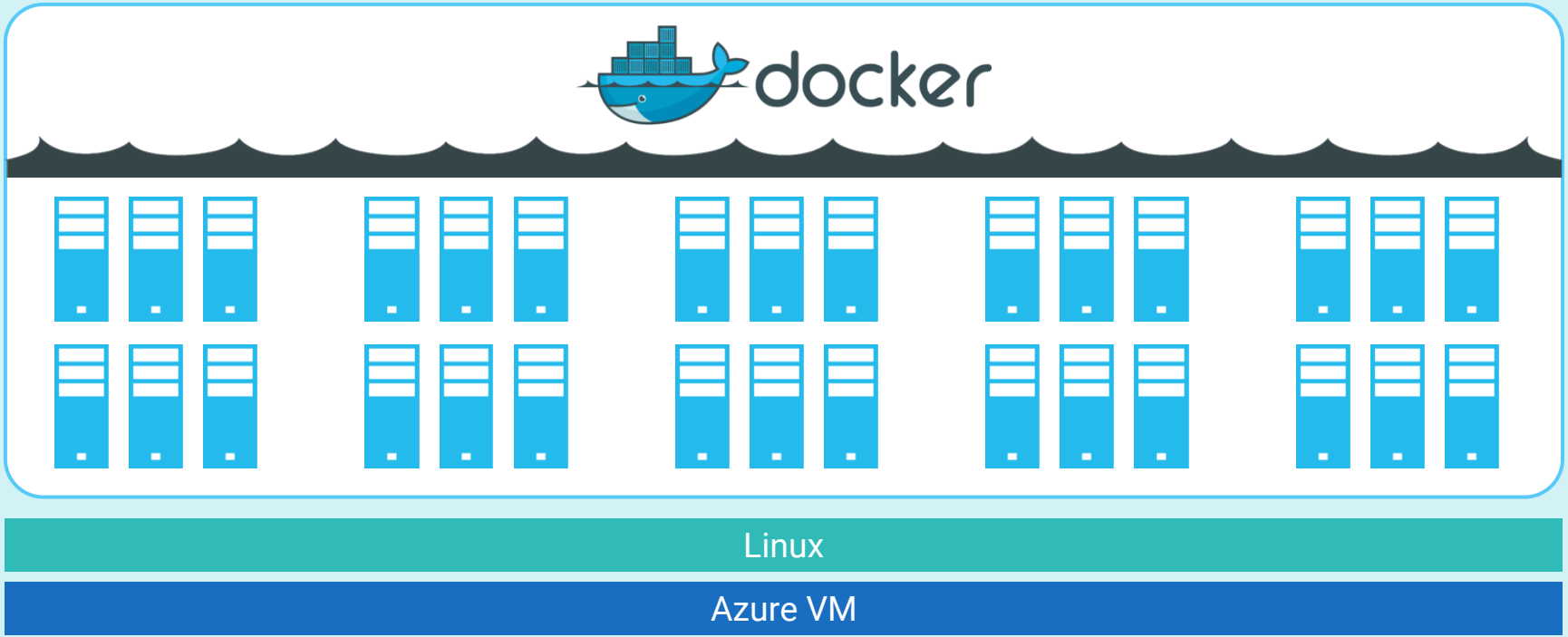

docker
docker
docker
docker
Linux
Linux
Linux
Linux
Azure VM
Azure VM
Azure VM
Azure VM
Hypervisor OS (Usually Linux-based)
Physical Datacenter Server (aka “Bare Metal” server)

Benefits of Containers

Benefits	
Lightweight	They are much lighter weight than virtual machines because they don't need to virtualize all computer hardware and don't need to run their own OS.
Share resources	Containers intelligently share OS resources while remaining isolated, allowing each one to focus exclusively on its own state.
Specialized	Containers only run the software components that they need to complete their task. Containers only do one thing.
Easily duplicated	A copy or image of a container can be easily downloaded and shared from computer to computer.
Prevalent and redundant	Containers are widely used in today's web architecture.

Containers and VMs

VMs and containers are used together. Usually an administrator will provision a powerful VM and then run many containers on it.

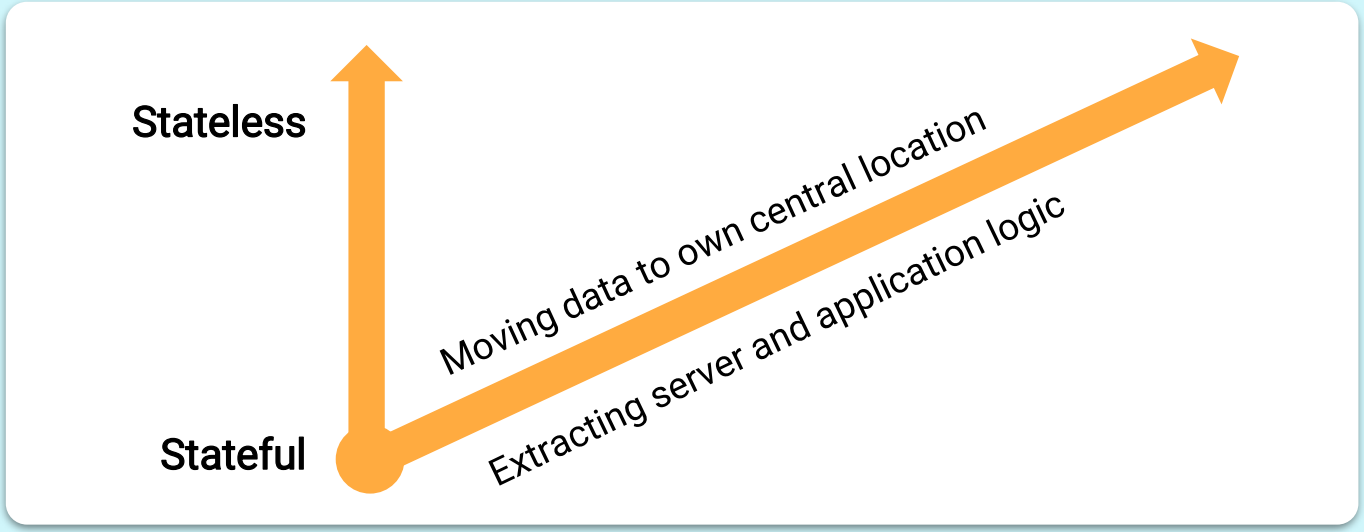


Using containers instead of VMs to run a LAMP server will immediately result in massive cost and operational savings. But, they still have to maintain their own data.

Stateless vs. Stateful Containers

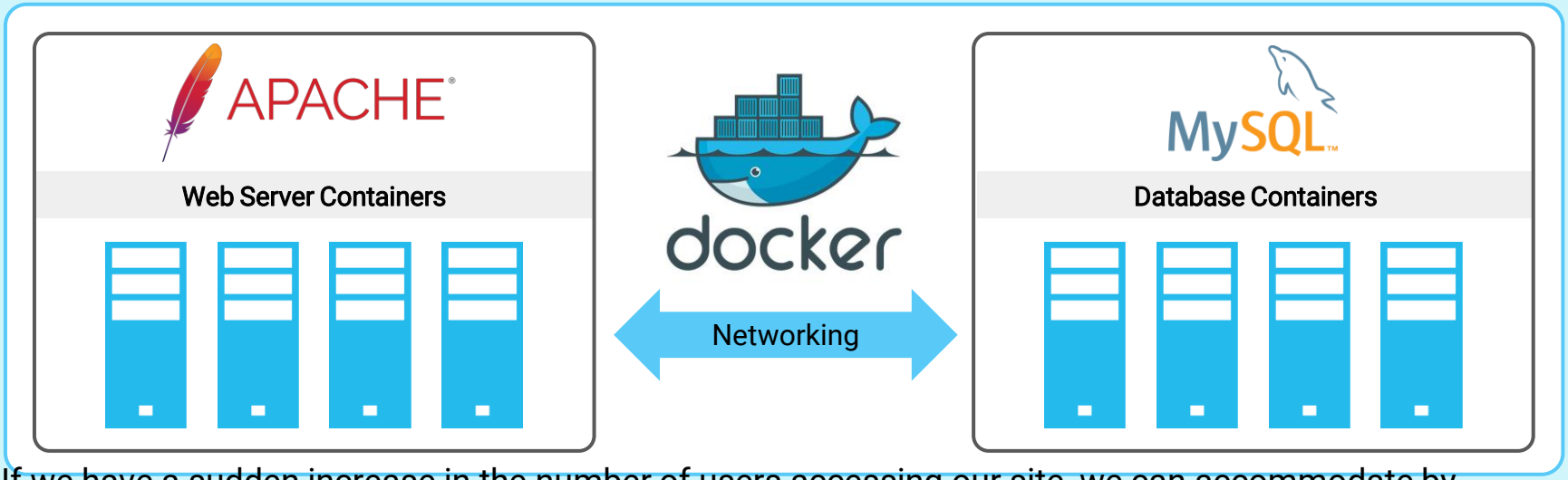
Containers are stateful. We can't safely destroy a container and replace it with a new one if it contains data that no other instance has.

Therefore, we need to make containers as stateless as possible, by moving data to its own central location and extracting server/application logic as much as possible.



As Stateless As Possible

In our LAMP container example, we can split the container into a set of multiple containers: one responsible for the database (MySQL) and one responsible for the web server (Linux/Apache).



If we have a sudden increase in the number of users accessing our site, we can accommodate by creating even more application/server containers and linking them to the database container.

Scaling

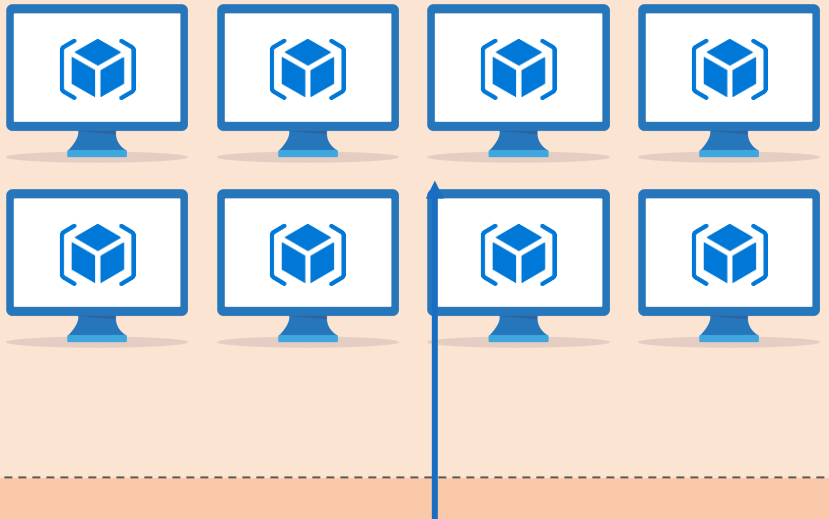
Creating more containers to handle additional load is called **horizontal scaling**. This is different from **vertical scaling**, where we simply make an existing machine more powerful by adding more RAM or CPU.

Vertical Scaling

Increase size of instance (RAM, CPU, etc.)

Horizontal Scaling

Add more instances



Horizontal Scaling

Horizontal scaling is vastly preferable to and more flexible than vertical scaling.

01

Horizontal scaling makes the system more resilient.

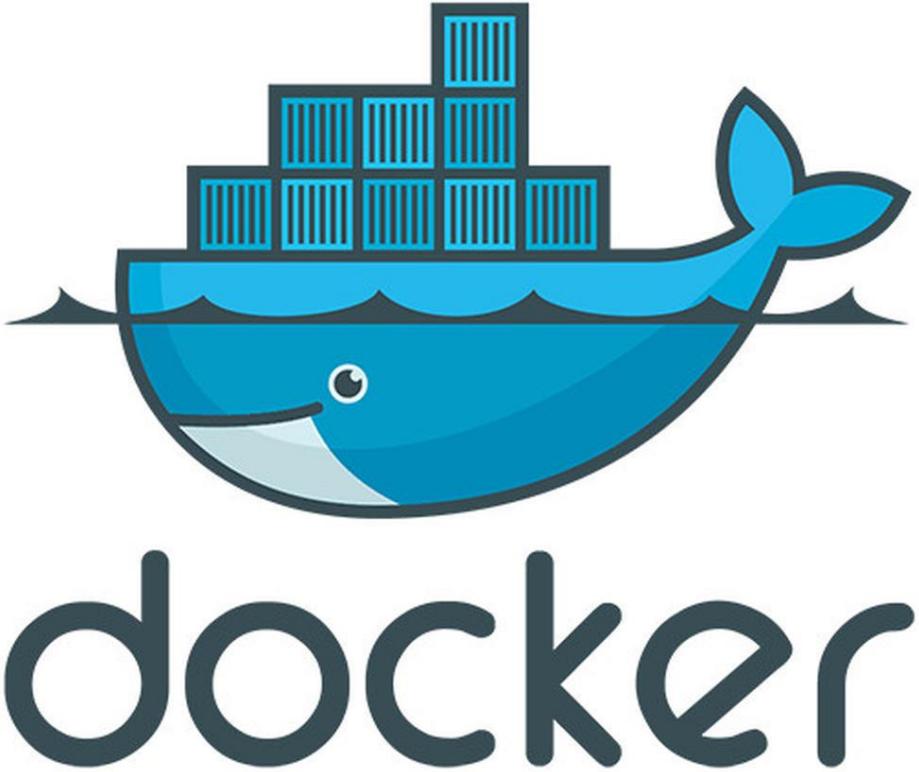
Multiple machines (or containers) prevent a single point of failure, as you can redirect requests to the other machines as needed.

02

Vertical scaling has a limit.

You can only add so much RAM and CPU to one computer before you run out of resources.

In real world practice, both vertical and horizontal scaling is used.



is the most common program used to manage containers.

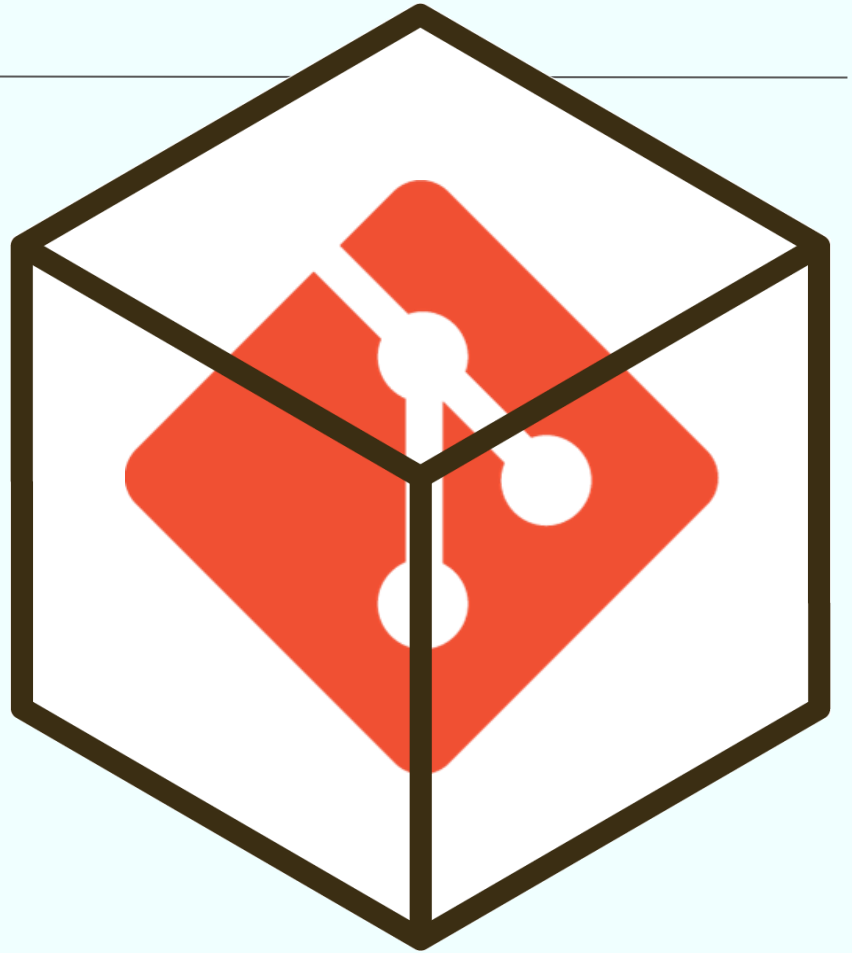
Docker

We can use Docker to distribute software, as an alternative to installing the software directly on a host machine.

Suppose we want to install a tool like Git on our VM. We can:

01 Install Git directly on our host.

02 Download a container that has Git inside of it and use Git from the container.

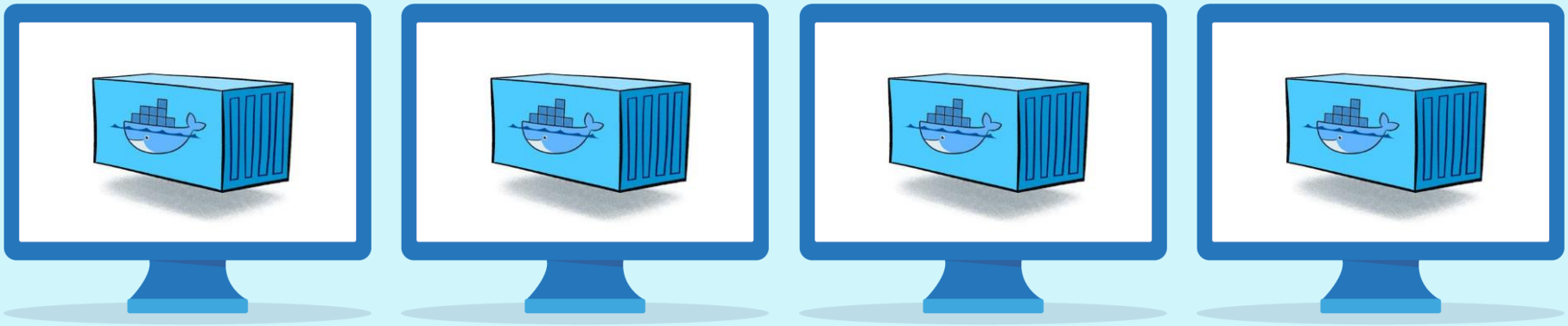


While downloading a container with Git adds a whole layer of indirection, it also eliminates platform and version issues.

Git Example

We can use the same container on any machine where you use Git, and it will work identically.

While we can't guarantee that software installed directly will behave identically across different machines, we can guarantee that the same container will.



This is usually not an issue with Git in particular,
but it is a significant source of bugs with other software.

Git Example

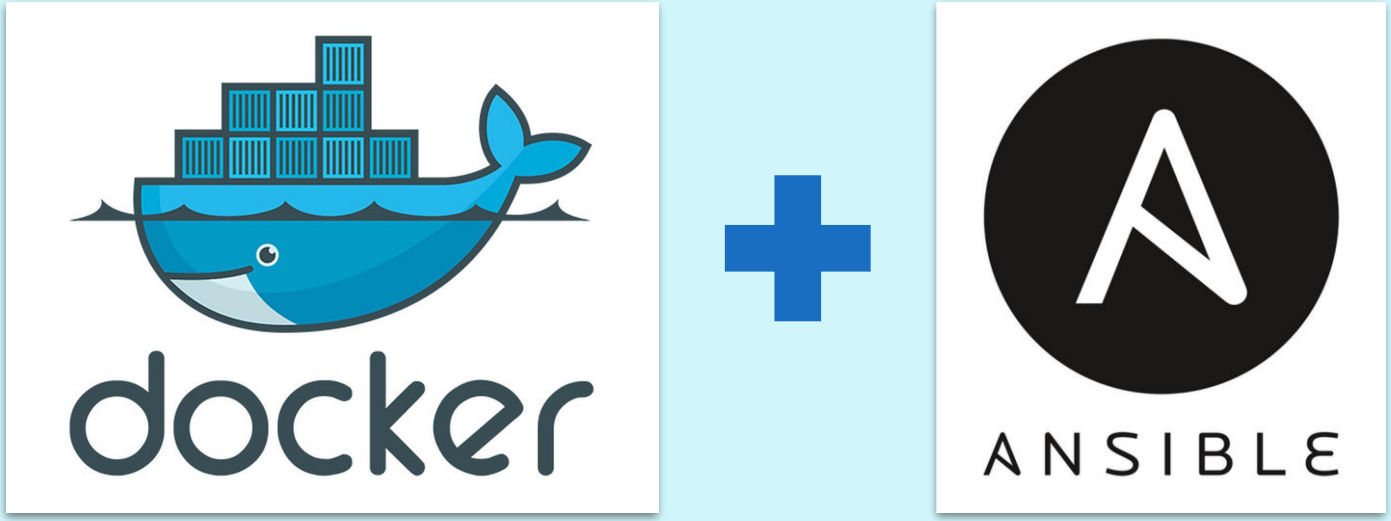
We can more easily install multiple versions of a tool side-by-side.

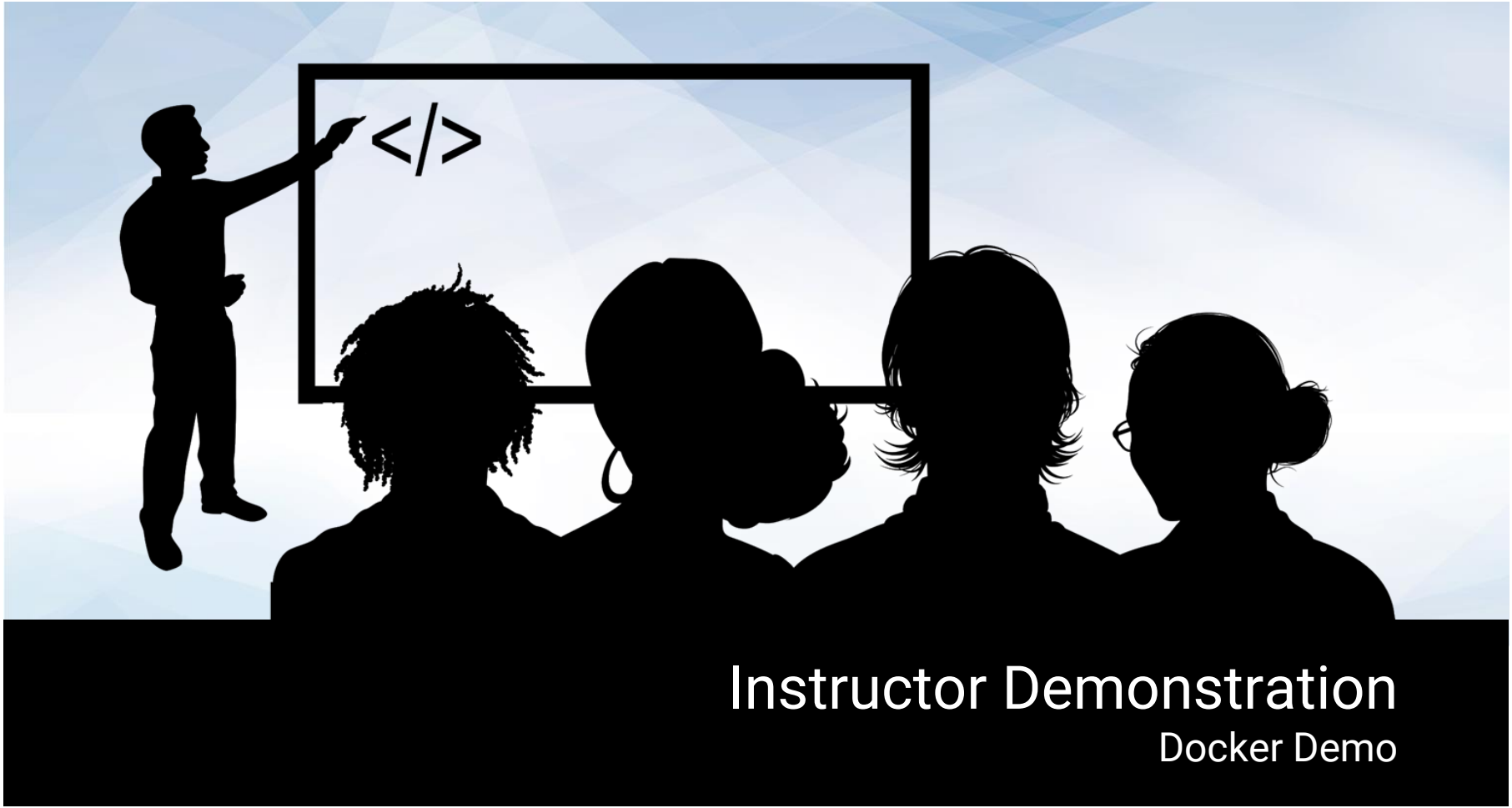
- Rather than managing several versions of a single software on a machine, we can have one container for each version we need.
- This is useful if we are working on several different projects at once, each needing a different version of the same software.
- This is actually an extremely common use case among developers, operations, and cloud engineering specialists.

Docker Demo Setup

In the following demo, we will use Docker to install Ansible, a provisioning tool, to ensure our provisioning scripts run identically everywhere they use them.

This further ensures our configurations will do exactly the same thing every time we run them, by eliminating as much variability between configurations as possible.





Instructor Demonstration

Docker Demo

Activity: Containers

In this activity, you need to configure your jump box to run Docker containers and install a container.

Suggested Time:
15 Minutes

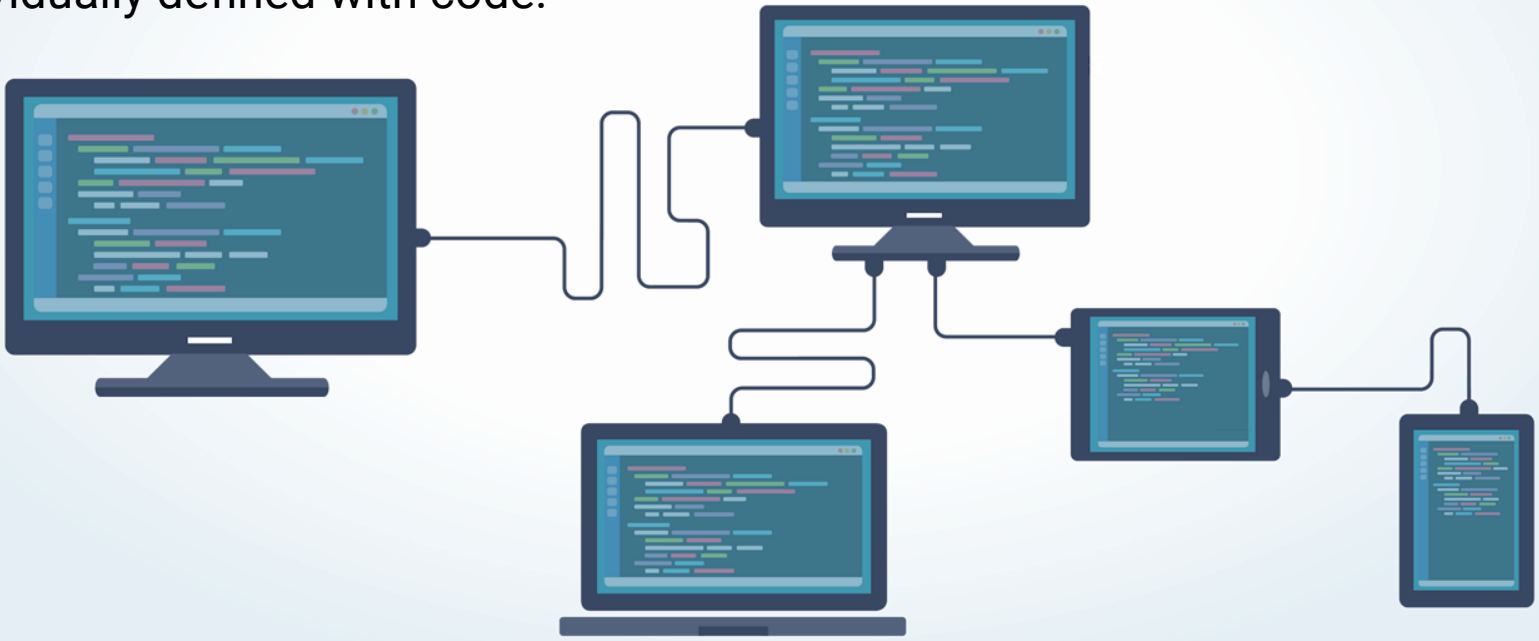
Time’s Up! Let’s Review.


Break

Now that we have our jump box running an Ansible Docker container, we will go over provisioners and configure this jump box to connect to other servers so it can configure them.

Infrastructure as Code

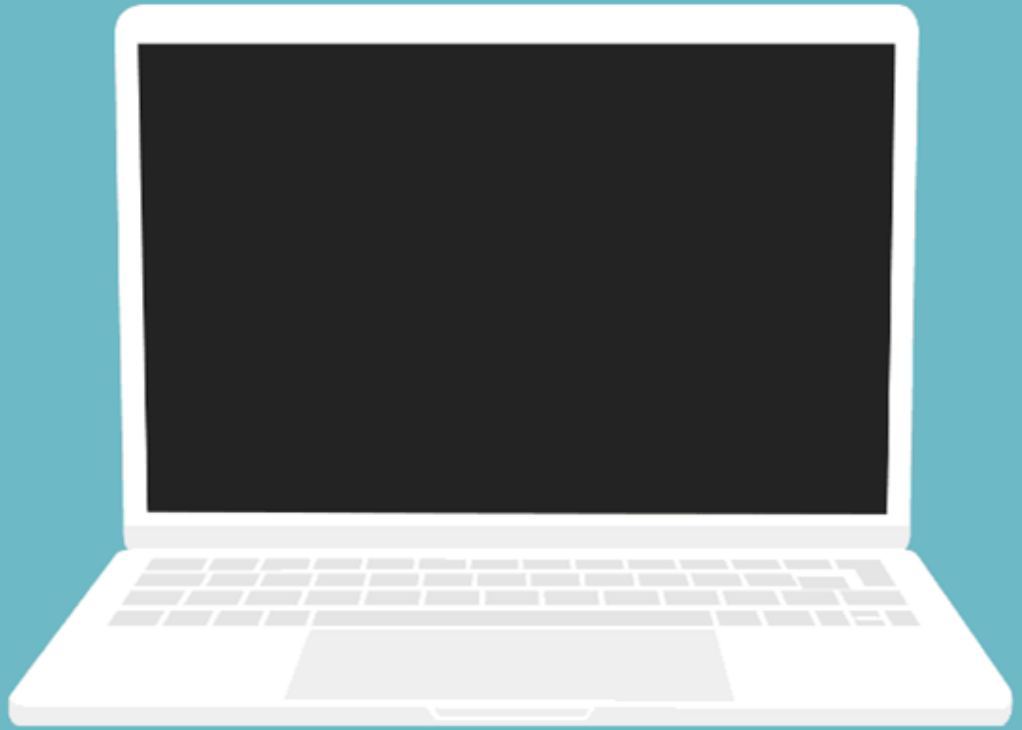
Infrastructure as code (IaC) is the concept of defining all of our equipment and network with code. When using virtual machines and containers, almost every server, database, workstation, and other component in your infrastructure can be individually defined with code.



Infrastructure as Code

When a particular piece of the infrastructure is needed, we can run the code that defines that thing and it will be up and running within a few minutes.

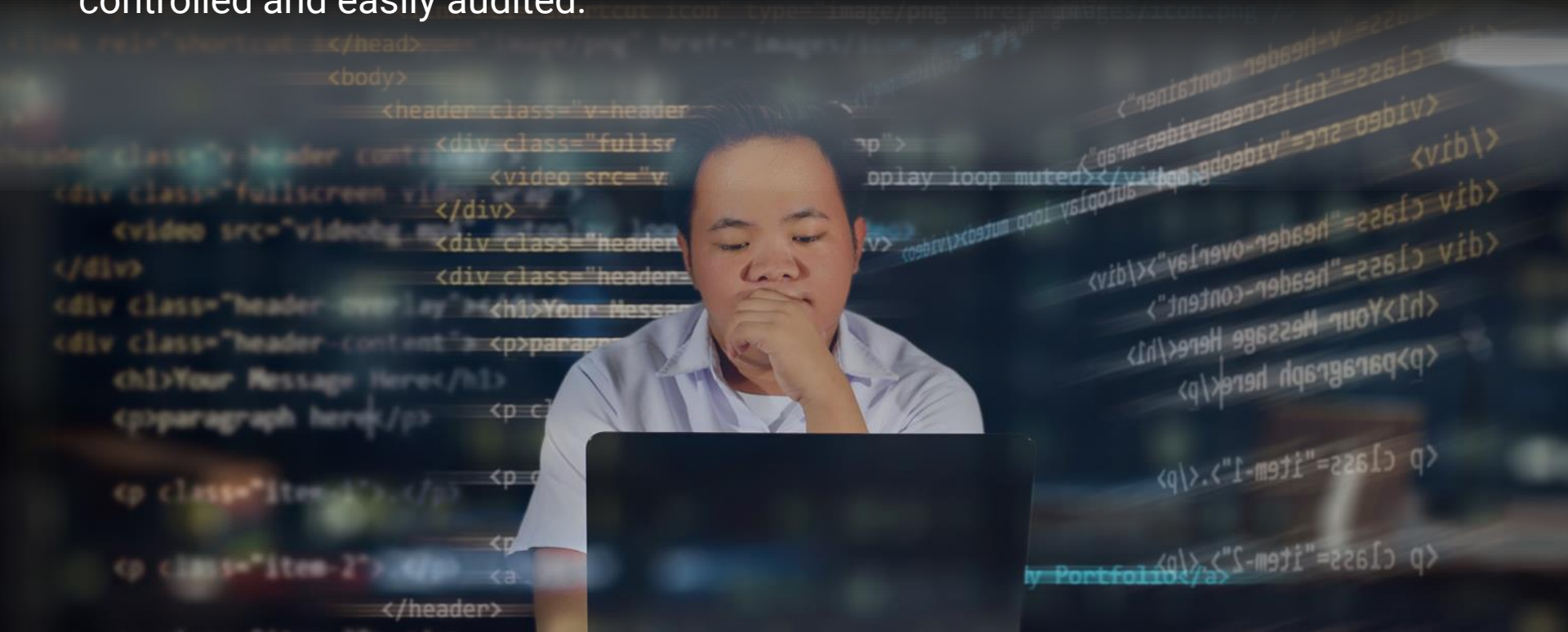
- IaC allows us to clearly build in security protocols from the ground up.
- If a server is found to be vulnerable, it's easy to change the code that created the server and build in a fix.



(Source)

IaC Change Management

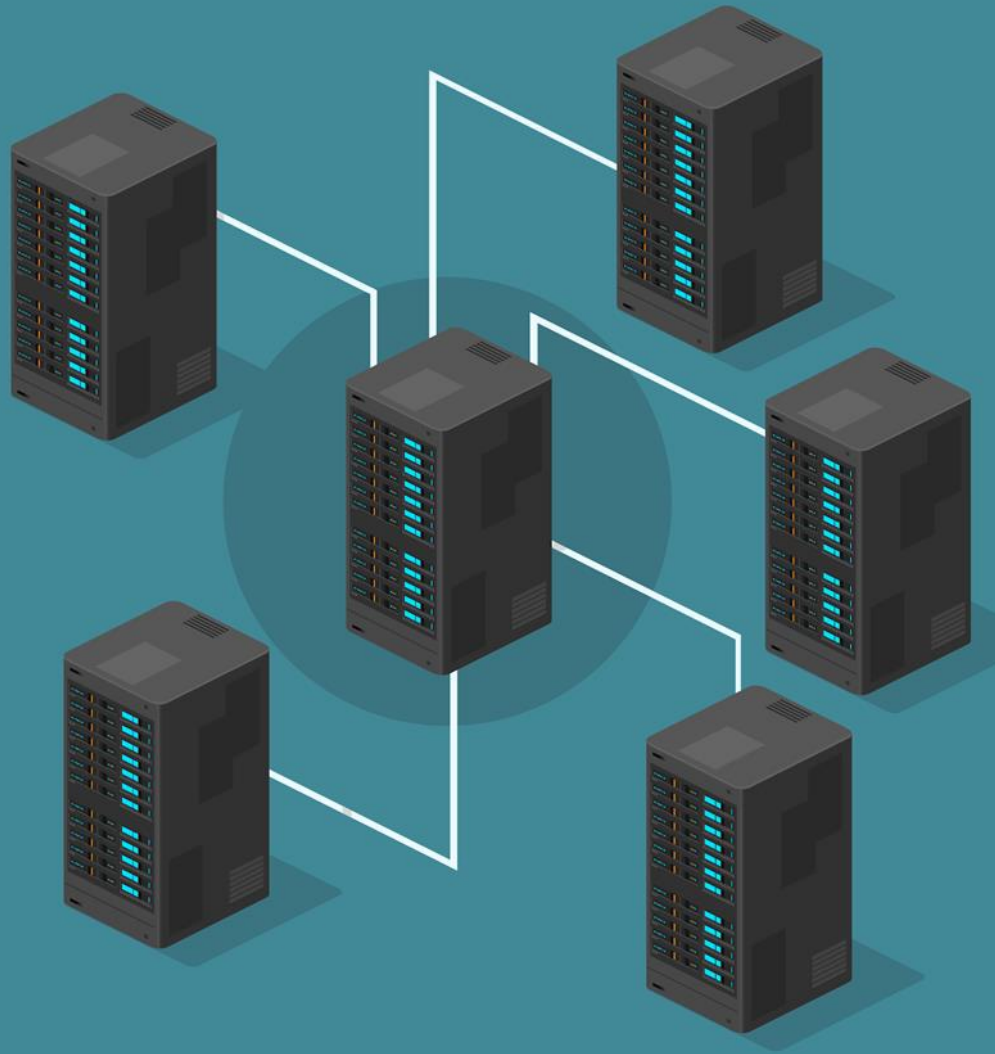
IaC is important for keeping track of the changes we've made. When we create code that contains the configuration of a server, that code can be version controlled and easily audited.



IaC Change Management

Rather than backing up the server and its settings, servers can send logs to a central database. This way, the only things needing backing up are small text files of the code that defines the servers.

- Code configuration changes can be deployed or reversed as needed.
- If an update causes a problem, we can use version control to reverse the code to its previous state and redeploy.



IaC Change Management

In order to see what changes are made to a server, we only need to look at what changes the code makes.

Often this code is written in a very easy-to-read language, so we only need minimal documentation to understand any given configuration.

Provisioners

A **provisioner** is a software application used in IaC setups used to make automated configuration changes to computers.

Provisioners focus on bringing a server to a certain state of operation.

Once the desired state of a server is documented with code, that code can be run on one server, 100 servers or 10,000 servers within a few minutes.

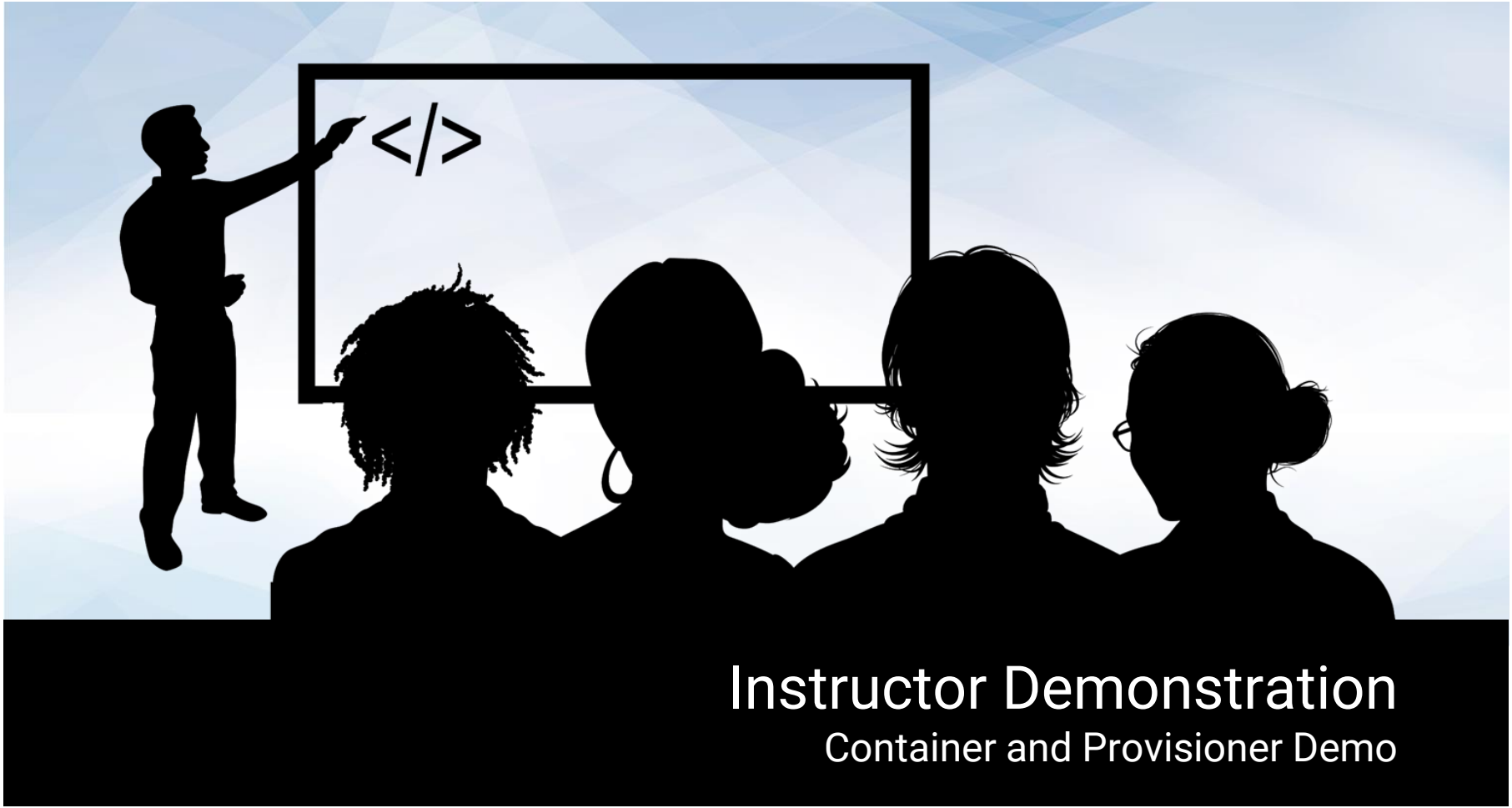
Changes made by a provisioner are created using text files, usually written in YAML or JSON.

Common provisioners include Ansible, Puppet, and Chef.

Container and Provisioner Demo

In the following demonstration, we will explore Docker and use our Ansible container to connect to a new VM.

This means we need to create an SSH key pair on the Ansible container and use that SSH `id_rsa.pub` file to create a new VM in Azure.



Instructor Demonstration

Container and Provisioner Demo

Activity: Provisioners

In this activity, you will launch a new VM from the Azure portal that can only be accessed using a new SSH key from the container running inside your jump box.

Suggested Time:
25 Minutes

Time’s Up! Let’s Review.

Daily Checklist

By the end of today, you should have completed the following critical tasks:

- Docker is installed and running on the jump-box.
- The cyberxsecurity/ansible Docker container is running on the jump-box.
- The security group has a rule that allows the jump-box SSH access to the vNet.
- An SSH key created from inside the Ansible container that has no password.
- The Web VM's password has been reset using the SSH key from the Ansible container.
- Ansible is able to make a connection to both Web VMs.

Don’t forget to power off your machine!


The End